PSO Week 7

CS448 Staff

Schema for Examples

We will consider example queries using the following schema:

```
Boats(bid, name, color)
Sailors(sid, sname, rating, age)
Reserves(sid, bid, day, rname)
```

where

- `sname` = sailor’s name
- `rname` = name of person who made reservation (might be different person than sailor)

Schema for Examples

We assume that

- `Reserves`
  - Each tuple is 40 bytes
  - 100 tuples per page
  - 1,000 pages
- `Sailors`
  - Each tuple is 50 bytes
  - 80 tuples per page
  - 500 pages

Motivating Example

SQL query

```
SELECT    S.sname
FROM      Reserves R, Sailors S
WHERE     R.sid = S.sid
          AND R.bid = 100 AND S.rating > 5
```

Relational Algebra

$$\Pi_{\text{sname}}(\sigma_{\text{bid}=100\wedge\text{rating}>5}(\text{Reserves}_{\bowtie_{\text{sid}=\text{sid}}}\text{Sailors}))$$

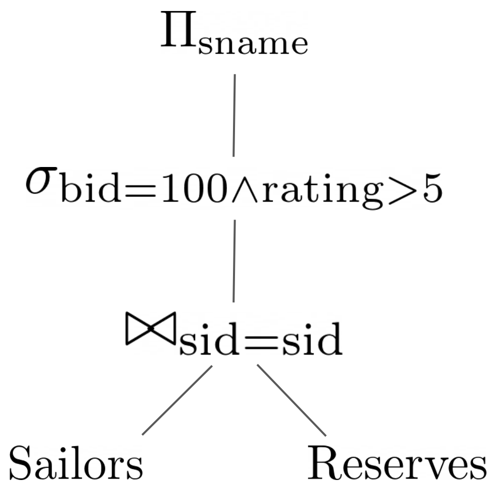
Motivating Example

SQL query

```
SELECT    S.sname
FROM      Reserves R, Sailors S
WHERE     R.sid = S.sid
          AND R.bid = 100 AND S.rating > 5
```

Relational Algebra

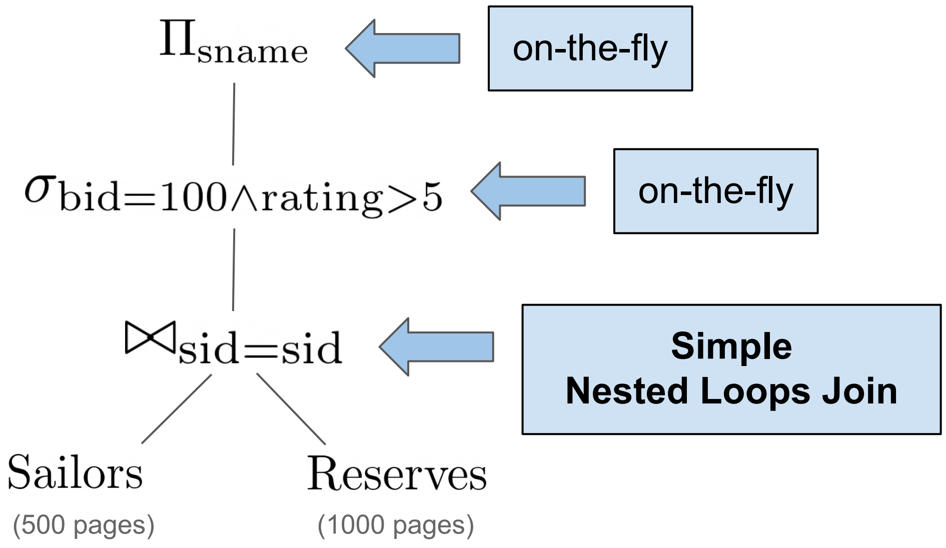
$$\Pi_{\text{sname}}(\sigma_{\text{bid}=100 \wedge \text{rating}>5}(\text{Reserves} \bowtie_{\text{sid}=\text{sid}} \text{Sailors}))$$



Π_{sname}
$\sigma_{\text{bid}=100 \wedge \text{rating}>5}$
$\bowtie_{\text{sid}=\text{sid}}$
Sailors
Reserves

Fig 1: Relational Algebra Tree

Query Evaluation Plan



Π_{sname}

on-the-fly

$\sigma_{\text{bid}=100 \wedge \text{rating}>5}$

on-the-fly

$\bowtie_{\text{sid}=\text{sid}}$

Simple Nested Loops Join

Sailors

(500 pages)

Reserves

(1000 pages)

Fig 2: Query Evaluation Plan

- **Cost**: 500 + 500*1000 I/Os
  - Pick R as outer join table = 500
  - Nested Loop Join = 500*1000
- **Misses opportunities:**
  - Selections could have been “pushed” earlier
  - No use is made of any available indexes
- **Goal of optimization:**
  - Find more efficient plans that compute the same answer

Alternative Plan 1: No Indexes

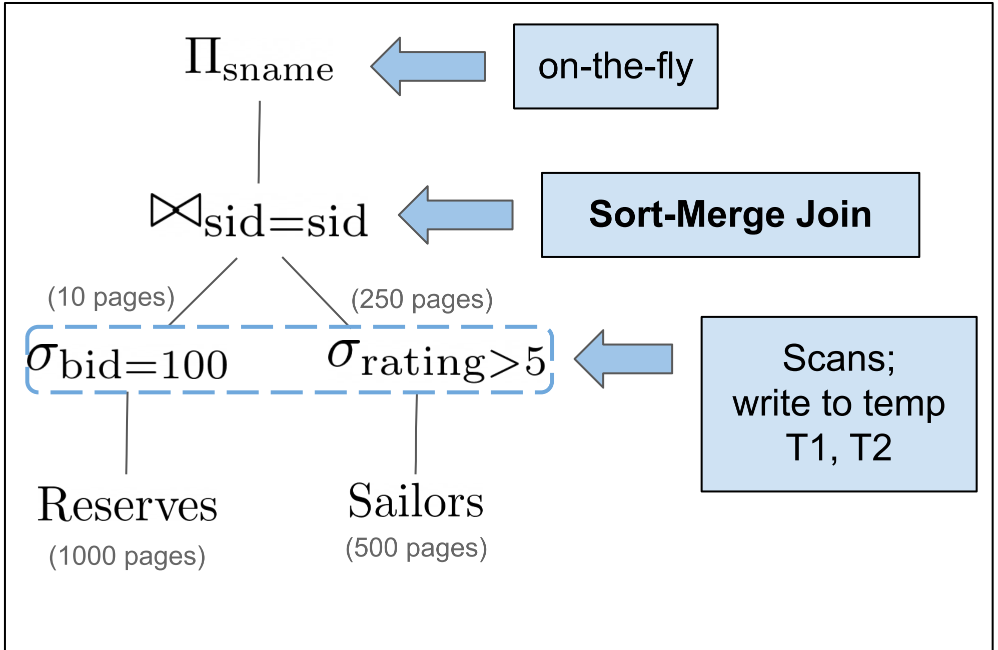

Π_{sname}
on-the-fly
$\bowtie_{\text{sid=sid}}$
Sort-Merge Join
(10 pages)
(250 pages)
$\sigma_{\text{bid=100}}$
$\sigma_{\text{rating>5}}$
Scans; write to temp T1, T2
Reserves
(1000 pages)
Sailors
(500 pages)

Fig 2a: Query Evaluation Plan (No Index, Sort-Merge)

- **Main difference:** push selects
- Assume we have 5 buffers
- **Cost**: 3560 I/Os
  - **Selects** (1760 I/Os)
    - Scan Reserves = 1000
    - Write T1 = 10
    - Scan Sailors = 500
    - Write T2 = 250
  - **Merge-Sort Join** (2300 I/Os)
    - Sort T1 = 2*2*10 (2 passes)
    - Sort T2 = 2*3*250 (3 passes)
    - Merge T1 and T2 = 10+250

Alternative Plan 1: No indexes

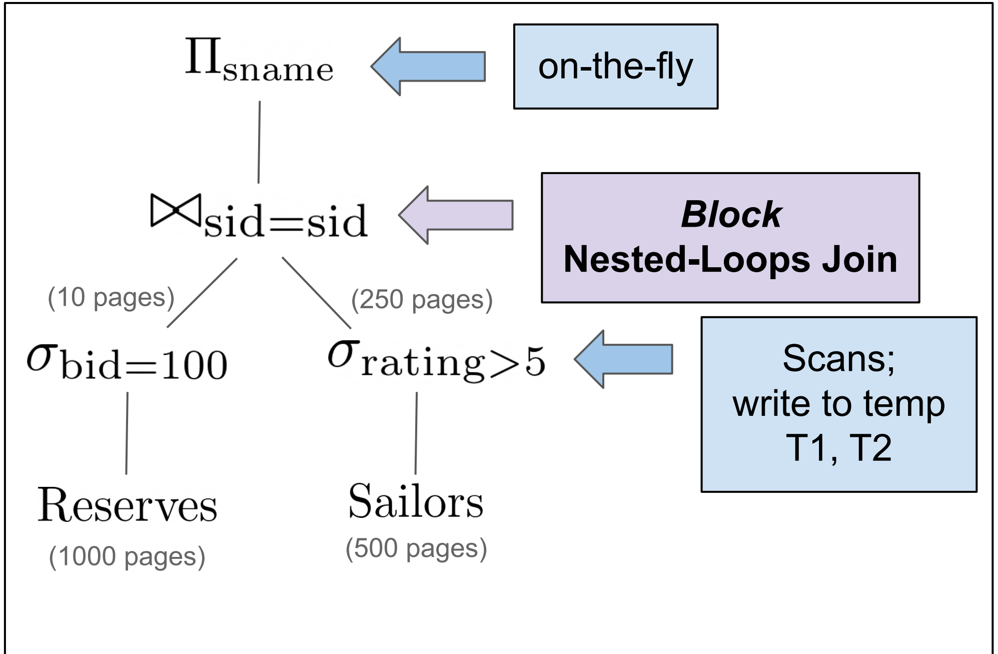


Fig 2b: Query Evaluation Plan (No Indexes, BNL)

- **Main difference:** push selects
- Assume we have 5 buffers
- **Cost**: 2770 I/Os
  - **Selects** (1760 I/Os)
  - **Block-Nested Loops Join** (BNL) (1100 I/Os)
  - Cost = Scan of outer + #outer blocks * scan of inner
    #outer blocks = ceiling(#pages of outer/blocksize)
  - Use T2 as Outer
  - 1 buffer holds T1, 3 buffer holds T2, 1 buffer output
  - #outer blocks = ceiling(250/3) = 84
  - Cost = 250 + 84*10 = 1090
  - Cost of flushing the buffer:10
  - Total I/O 1100 = 1090+10

Alternative Plan 1: No Indexes

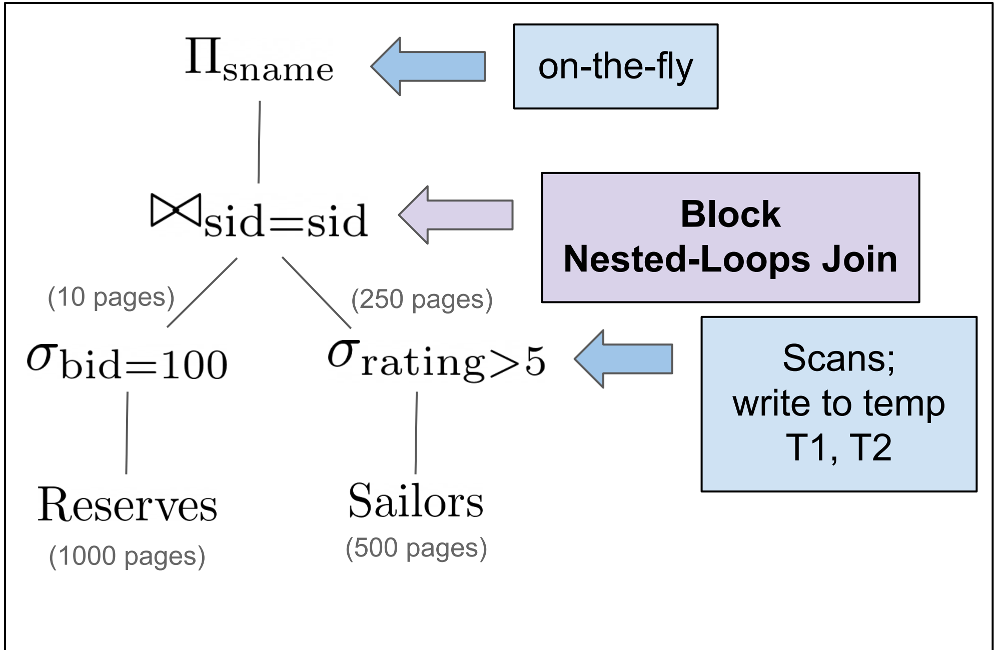
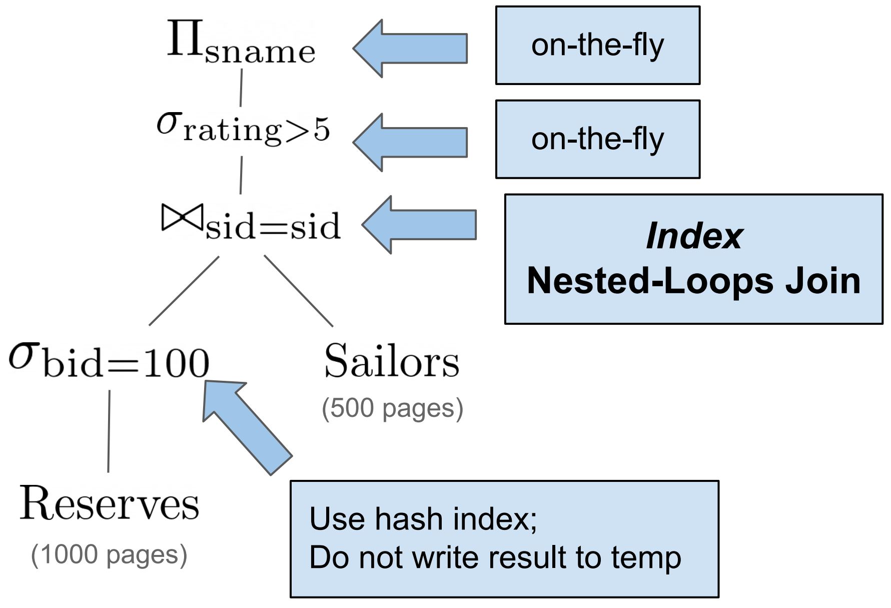


Fig 2b: Query Evaluation Plan (No Indexes, BNL)

- **Main difference:** push selects
- Assume we have 5 buffers
- **Cost**: 2700 I/Os
  - **Selects** (1760 I/Os)
  - **Block-Nested Loops Join** (BNL) (1100 I/Os)

- *If we "push" projections*
  - **T1** has only `sid`
    - fits in 3 pages
  - **T2** has only `sid` and `name`
  - **BNL cost:** *under 250 pages*
  - **Cost total:** < 2000 I/Os

Alternative Plan 1: With Indexes



Π_{sname}

on-the-fly

$\sigma_{\text{rating}>5}$

on-the-fly

$\bowtie_{\text{sid}=\text{sid}}$

Index
Nested-Loops Join

$\sigma_{\text{bid}=100}$

Sailors
(500 pages)

Reserves
(1000 pages)

Use hash index;
Do not write result to temp

Fig 3: Query Evaluation Plan (With Index)

- Clustered index on `bid` of Reserves
  - # selected tuples: 100,000/100 = 1000 tuples, on 1000/100 = 10 pages
- **INL** with pipelining (projecting out unneeded fields does not help)
- Join column `sid` is *key* for Sailors

- **Cost**: 1210 I/Os
  - Select Reserve tuples (10 I/Os)
    - For each, get matching Sailors tuple (1000*1.2 I/Os)

Summary

- A query is evaluated by converting it to a tree of operators and evaluating the operators in the tree
- Two parts for optimizing queries
  - Consider a set of alternative plans
    - Must prune search space; typically left-deep plans only
  - Must estimate cost of each plan that is considered
    - Estimate size of result and cost for each plan node
- Must understand query optimization in order to fully understand the performance of a given database design (relations, indexes) on a workload (set of queries)

Reference: *Database Management Systems, 3rd edition, by Ramakrishnan and Gehrke*